Управление задание:

Посылка сигналов процессу:

- Ctrl + C - SIGNIN приводит к прерыванию
- Ctrl + \ - SIGQUIT завершение процесса и создание дампа памяти
- Ctrl + Z - SIGSTOP остановка

Автоматизация работы:

at - создание задачи
atq - очередь задач (выведет список всех запланированных задач)
atrm - удаление задания
at now + n minutes - запустить команду через n минут
batch - выполнит задание, когда будут доступны ресурсы
Команды можно добавлять в файл, чтобы не вводить их огромное кол-во раз

Сценарий (Scripts)

Командный процесс называется sh (shell - оболочка)/bash/dash/zsh
Все языки программирования, используемые в командной строке - интерпретируемые.
Все сценарии начинаются со строки '#!/bin/bash'
Запуск команды происходит через команду 'bash$'
sh -x - позволяет увидеть пошаговое выполнение скрипта
sh -v - позволяет увидеть больше информации

Системные переменные

Создание переменной:
bash$ SCRIPT_PATH = ...
Чтобы использовать её нужно прописать команду:
bash$ export NAME

Язык программирования

- комментарий
$? - код завершения последнего процесса
local - ключевое слово для переменной, которая доступна только в момент работы скрипта
$name_var - обращение к переменной

Арифметические операции

expr - команда, которая высчитывает результат математического

выражения
var = $((n + m)) - команда для подсчёта суммы чисел n и m
if [$usl1 == ...]; then ... ; fi - пример условного выражения

Примеры команд сравнения чисел:

- -eq - равно
- -ne - неравно
- -lt - меньше
- -le - меньше или равно
- -gt - больше
- -ge - больше или равно

Если строки:

- = - равно
- != - не равно
- < - Первая строка отсортирована перед второй
- > - Первая строка отсортирована после второй

Циклы и функции
for ((<начальное значение>; <условие завершение>; <инкремента/декремента>)); do ...; done - цикл
seq <старт> <шаг цикла, если чисел 3> <конец> - создание списка от старта до конца с шагом

select <переменная> in <список>; do ... ; done - если переменная есть в списке, то...
case <переменная> in - аналог switc/case из Си
func_name(){ ... } - тип функции
side effect (побочный эффект) - эффект, который меняет глобальную переменную

Типы компиляторов для оптимизации:

- оптимизация под типы приложения
- процессорные оптимизации
- дополнительные оптимизации

GCC - GNU Compile Collection
gcc - GNU C compiler

Опция оптимизации:

- /0d - отключение оптимизации (Расширения файла по мере компиляции: .cpp -> .o -> .elf)
  символ - переменный процесс или ресурс, доступный в исполняемой форму
- /01 - использует методы оптимизации, который не увеличивает исполняемого кода (.elf)
- /02 - ускоряет скорость работы кода, но немного увеличивает размер исполняемого кода
- /03 - увеличивают размер исполняемого кода, но мощно ускоряет скорость работы компилятора
- /04 - экспериментальный метод оптимизации